**AFRL-IF-RS-TR-2001-288**
**Final Technical Report**
**January 2002**

# OMNISLEUTH

**Odyssey Research Associates**

*APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.*

**AIR FORCE RESEARCH LABORATORY**
**INFORMATION DIRECTORATE**
**ROME RESEARCH SITE**
**ROME, NEW YORK**

20020507 094

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2001-288 has been reviewed and is approved for publication.

APPROVED: John Feldman

JOHN FELDMAN
Project Engineer

FOR THE DIRECTOR:

WARREN H. DEBANY, Technical Advisor
Information Grid Division
Information Directorate

If your address has changed or if you wish to be removed from the Air Force Research Laboratory Rome Research Site mailing list, or if the addressee is no longer employed by your organization, please notify AFRL/IFGB, 525 Brooks Road, Rome, NY 13441-4505. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

| | |
|---|---|
| **REPORT DOCUMENTATION PAGE** | *Form Approved*<br>*OMB No. 0704-0188* |

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

| | | |
|---|---|---|
| **1. AGENCY USE ONLY** *(Leave blank)* | **2. REPORT DATE**<br>JANUARY 2002 | **3. REPORT TYPE AND DATES COVERED**<br>Final Oct 99 - Jul 00 |

**4. TITLE AND SUBTITLE**

OMNISLEUTH

**5. FUNDING NUMBERS**

C - F30602-99-C-0009
PE - 33140F
PR - 7820
TA - 09
WU - P1

**6. AUTHOR(S)**

Matt Stillerman and David Rosenthal

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**

Odyssey Research Associates
Cornell Business & Technology Park
33 Thornwood Drive, Suite 500
Ithaca New York 14850-1250

**8. PERFORMING ORGANIZATION REPORT NUMBER**

N/A

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**

Air Force Research Laboratory/IFGB
525 Brooks Road
Rome New York 13441-4505

**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

AFRL-IF-RS-TR-2001-288

**11. SUPPLEMENTARY NOTES**

Air Force Research Laboratory Project Engineer: John Feldman/IFGB/(315) 330-2664

**12a. DISTRIBUTION AVAILABILITY STATEMENT**

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

**12b. DISTRIBUTION CODE**

**13. ABSTRACT** *(Maximum 200 words)*

OmniSleuth provides an integrated environment for investigation of cyber attacks, including the gathering, preservation, organization, and analysis of evidence. OmniSleuth gives an investigator access to remote computers by supporting the deployment of investigative agents to those computers. An agent is a program that is run on a remote, networked computer by sending the program executable to that remote machine. Typically, agents communicate with one another and can be controlled remotely. OmniSleuth investigative agents observe the state and activity of the host platform and can report on it. The investigator can control these remotely running agents. The reports of individual observations, called events, may be retrieved by the investigator and preserved in a database. The events in this database may then be organized and associated with parts of a hypothesized crime theory, either in support or as a refutation of that theory. OmniSleuth's agent infrastructure is based on the result of an earlier project, the Secure Intrusion Detection Framework (SIDF) which enables the secure deployment of intrusion detection functionality to a network in the form of software agents.

**14. SUBJECT TERMS**

Network Forensics, Secure Agents, Remote Forensics, Computer Forensics

**15. NUMBER OF PAGES**

36

**16. PRICE CODE**

| | | | |
|---|---|---|---|
| **17. SECURITY CLASSIFICATION OF REPORT** | **18. SECURITY CLASSIFICATION OF THIS PAGE** | **19. SECURITY CLASSIFICATION OF ABSTRACT** | **20. LIMITATION OF ABSTRACT** |
| UNCLASSIFIED | UNCLASSIFIED | UNCLASSIFIED | UL |

Standard Form 298 (Rev. 2-89) (EG)
Prescribed by ANSI Std. 239.18
Designed using Perform Pro, WHS/DIOR, Oct 94

# Table of Contents

1 Summary ........................................ 1
2 Introduction ........................................ 2
    2.1 Scope ........................................ 3
    2.2 Objectives ........................................ 3
    2.3 What Follows ........................................ 4
3 Requirements for Network Investigations ........................................ 4
    3.1 Special Considerations in Network Environment ........................................ 4
        3.1.1 Ephemeral State and Events ........................................ 4
        3.1.2 Scale and Complexity of the Evidence ........................................ 5
        3.1.3 Impact of Investigation on Critical Infrastructure ........................................ 6
    3.2 Non-disclosure ........................................ 6
        3.2.1 Keeping Suspects in the Dark ........................................ 6
        3.2.2 Legitimate Privacy Concerns ........................................ 6
    3.3 Integrity ........................................ 7
        3.3.1 Integrity of the Investigative Record ........................................ 7
        3.3.2 Preservation of Raw, Un-interpreted State ........................................ 7
    3.4 Availability – Ensuring Timely Investigation ........................................ 8
4 OmniSleuth Capabilities ........................................ 8
    4.1 Distributed search across many hosts ........................................ 9
    4.2 Specialized forms of observation ........................................ 9
    4.3 Organization of Findings ........................................ 10
    4.4 Increasing Statefulness ........................................ 11
    4.5 Alternative Views of the Data ........................................ 12
    4.6 Transition to Conventional Investigation ........................................ 12
5 Architecture ........................................ 13
    5.1 Major OmniSleuth Components ........................................ 14
    5.2 Event Subsystem ........................................ 14
    5.3 OmniAgent, base class ........................................ 15
    5.4 Notions of Identity for the AG's and the Agents ........................................ 16
    5.5 Theory of the Crime Interface ........................................ 16
    5.6 Timeline Tool ........................................ 19
6 Scenario ........................................ 19
    6.1 The Scenario ........................................ 20
    6.2 Testing Our Assumptions Against the Scenario ........................................ 21
    6.3 OmniSleuth Capabilities Used in the Scenario ........................................ 22
7 Conclusions and Recommendations ........................................ 23
    7.1 Accomplishments ........................................ 23
    7.2 Current Status ........................................ 24
    7.3 Future Directions ........................................ 24
8 References ........................................ 25

## List of Figures

Figure 1. Agent Deployment in OmniSleuth 14

Figure 2. The Theory of the Crime Interface 17

Figure 3. The Timeline Tool 19

# 1 Summary

Computers are being used with increasing frequency to commit crimes. In addition to computer break-ins, criminals are employing computers to commit theft, fraud, and other "traditional" crimes. Computers are also used incidentally in many activities and thus could contain evidence of a crime. Due to the trend towards increasing reliance on computers in all aspects of our civilization, crimes in which computers play some role are expected to become more common in the near future. Therefore, the need to investigate such crimes, including the collection of evidence from computers, will be more important.

Current computer forensic tools are generally geared toward collecting and analyzing data from some storage device. However, there is little support for the kind of systematic investigation that involves multiple platforms that are heterogeneous, mission critical, and geographically dispersed. This description could easily apply to the computing resources of any large organization. Thus, some advance in investigative technique and tools will be necessary for investigations to become practical in this kind of setting.

This report describes our work on *OmniSleuth* a prototype investigative framework and tool [Rei00]. *OmniSleuth* is intended to facilitate investigations in a network environment. It provides an integrated environment for investigation, including the gathering, preservation, organization, and analysis of evidence. Our goal for the project was to study some issues that will arise in network investigations, and to demonstrate the potential for addressing those issues with an integrated environment.

OmniSleuth gives the investigator access to remote computers by supporting the deployment of *investigative agents* to those computers. An agent is a program that is run on a remote, networked computer by sending the program executable to that remote machine. Typically, agents (running agent programs) communicate with one another and can be controlled remotely. OmniSleuth investigative agents observe the state and activity of the host platform and can report on it. The investigator can control these remotely running agents. The reports of individual observations, called *events*, may be retrieved by the investigator and preserved in a database. The events in this database may then be organized and associated with parts of the *crime theory* that they support or refute.

Security, another important aspect of OmniSleuth, supports the integrity and completeness of the investigative record. We are also concerned that the privilege inherent in a powerful investigative system like OmniSleuth not be abused. Access control and mechanisms to enforce accountability will mitigate this threat.

OmniSleuth's secure agent infrastructure is based on the result of an earlier Odyssey Research Associates (ORA) project, the *Secure Intrusion Detection Framework* (SIDF). This framework enables the secure deployment of intrusion detection functionality to a network in the form of software agents. Within SIDF (and OmniSleuth), secure communications and authentication are accomplished by using PKCS-11 compliant cryptographic tokens, giving a high degree of assurance for security properties.

Although our original idea that there should be some valuable synergy between intrusion detection functions and investigative functions has so far not been demonstrated by our work, the idea is still very attractive.

## 2 Introduction

This is the final report of the project entitled *Omni-Sleuth - Computer Forensic System* of Odyssey Research Associates, contract number F30602-99-C-0009. This work was performed under the direction of the Air Force Research Laboratory, Rome Site.

The primary goal of this project was to explore some novel ideas about how computer investigations might be conducted in a network environment. The work is motivated, in part, by the need for investigative techniques to "scale up" to contemporary and near-future networks. Current *computer forensic* investigative practice involves sequestering the primary physical storage media and examining copies of that media off-line. This painstaking process, requiring special equipment and skills, is intended to produce evidence suitable for use in court by virtue of its manifest integrity.

The nature and scale of computer networks is changing rapidly – they are becoming larger, more heterogeneous, and more ubiquitous. This evolution poses new challenges for investigators in performing routine investigations involving computers. The most significant challenges that we envision are these:

**Complexity:** A network-level investigation could potentially produce a virtual avalanche of data about the state of the target computers. Analysis of that data may require distinct interpretations of similar data when collected on different hosts, or in different circumstances.

**Criticality:** It may not be acceptable to shut down and sequester a target host without compelling evidence of necessity due to the harm, inconvenience, and loss of service that such a disruption would cause to innocent bystanders and actually innocent suspects.

**Integrity:** The integrity and correctness of investigative results should be unimpeachable. Yet, by not preserving the primary evidence (the original physical media), the results may be subject to doubts that cannot be resolved, since the *complete* analysis cannot be repeated. Therefore, the evidence gathered must be preserved in a tamper-resistant manner, together with sufficient information about its provenance and meaning, to deter any such arguments.

**Scale:** The resources necessary to investigate all relevant target computers in a large network using current forensic methods would often be impractical, except in very "high value" investigations.

**Ubiquity:** Computers are involved *incidentally* in many aspects of our lives, and so would naturally be relevant to investigations of even "ordinary" crimes (or misdeeds within an organization). This factor drives up the demand for this kind of investigation.

To meet these challenges, we envision an integrated system for gathering evidence on a network of computers, and for organizing, displaying, and analyzing that evidence.

## 2.1 Scope

The scope of this effort is to address the needs of criminal investigators and organizational security officers who are investigating activities that involve computers and networks either centrally or incidentally. We consider only situations where the information systems that constitute the "scene" of the investigation are under the control of the investigator's organization. Thus, we assume that the organization can *predeploy* certain software to the hosts under its control that might someday be the target of an investigation. This assumption might be appropriate for investigations on an Air Force base or in a large corporation. On the other hand, most of the techniques studied here are not intended for investigating criminal activity on the Internet at large.

Evidence that will eventually be presented in court must meet strict standards intended to ensure its integrity. In this effort, we are concerned with the integrity of evidence that is gathered over the network. We studied means for enhancing and preserving confidence in its integrity. However, we do not make any claims about the adequacy of such measures for courtroom admissibility. Eventually, we believe, courts must clarify standards for admissibility for such evidence.

Both the forensics and intrusion detection research communities are studying how to detect the traces of criminal activity on computer systems in isolation. This project is concerned only with the special needs and consequences of deploying such detection algorithms remotely over a network. The project scope, therefore, did not include development of new host-based detection algorithms. Rather, it assumed that such algorithms will be available from the industrial and research communities.

## 2.2 Objectives

The objectives of this project are

- to articulate some requirements for extending the reach of crime investigation to networked environments,
- to delineate the core capabilities, functionality, and architecture of an integrated network investigation tool, and
- to demonstrate the feasibility of such a tool by building and demonstrating a prototype.

Our prototype network investigation tool is called *OmniSleuth*.

### *2.3* What Follows

Section 3, *Requirements for Network Investigation*, describes our assumptions about the nature of investigations generally and in the anticipated network environment, and the consequent requirements on OmniSleuth.

Section 4, *OmniSleuth Capabilities*, describes the OmniSleuth system in terms of its capabilities (or functionality) and relates these to the requirements.

Section 5 describes the architecture of OmniSleuth, including some features inherited from the SIDF infrastructure.

We guided aspects of our design and development with a *scenario* of a hypothetical network investigation. This scenario is described in detail in Section 6. In particular, motivation for specific system capabilities within the scenario is pointed out.

Section 7, *Conclusions and Recommendations,* describes our principal accomplishments and the status of the software. More detailed status information is in the Software Users Manual. Section 7 also contains possible directions for future work on OmniSleuth.

## 3 Requirements for Network Investigations

We made some assumptions about the task faced by *future* investigators working in a network environment. In part, these assumptions are the result of extrapolation from current requirements and current practice. Other assumptions are implicit in the scope of the project (see Section 2.1). These assumptions dictated (or suggested) capabilities that an investigator would need in order to be effective. The capabilities have to do with the networking environment, as well as requirements for non-disclosure, integrity, and availability.

### 3.1 Special Considerations in Network Environment

The following subsections describe special considerations for investigations in a network environment.

#### 3.1.1 Ephemeral State and Events

Investigators will examine the *state* of the target computers for traces of the activities of suspects. Often, however, aspects of the state that would be of great interest are short-lived: they may be overwritten by new state in the course of normal operation, or they may be tampered with by suspects trying to cover their tracks. If it were available, the investigator might also make good use of some information about events that have occurred, such as the arrival of certain network packets. In either case, this data will be lost if the investigator was not "looking" when the state existed or the event occurred.

We explored an approach to the problem of ephemeral state in which the investigative system can be tasked to observe and record such information proactively, before it is needed. This recording is under the control of the investigator and might be initiated either as part of a general policy of the organization or as part of an investigation. We call this "increasing the statefulness" of the target computer.

Increasing statefulness poses two challenges that are addressed in the architecture and design of OmniSleuth:

- How to do it inexpensively enough, given that most of the data recorded may not ever be used.
- How to handle that data so that when it is requested, it can be supplied with very high assurance of integrity, even though the original observation and the request are separated in time.

### 3.1.2 Scale and Complexity of the Evidence

When conducting an investigation in a digital environment, many assumptions and structures will be explicit that in a corresponding "ordinary" investigation would be taken for granted. For instance, the presence of a fingerprint on some piece of furniture in a room corroborates that the suspect (say) entered that room. However, a careful analysis shows

- that the phrase "entered that room" is a shorthand for a complex ensemble of events and alternatives that are hypothesized,
- that the fingerprint is direct evidence of just one small part of that hypothesis,
- that the "corroboration" is via some fairly fancy reasoning involving lots of common sense knowledge about the parts of the hypothesis, as well as the observed facts, and
- that we must assume the fingerprint expert searched thoroughly and handled and identified the fingerprint competently and is reporting honestly about his activities.

A similar constellation of issues will arise in computer investigations. However, we can expect these to be much more explicit for several reasons: First, the knowledge necessary to tie together the observed facts and the parts of the hypothesis will not be "common sense." Rather, we can expect this knowledge to be quite arcane (e.g. to depend on versions of the operating system and quirks of particular application programs). Second, the correctness of the observations is established and maintained by explicit bookkeeping and security measures that allow the observation to be traced back to a "correct" observer program (an agent) and an authorized investigator. In order to do this bookkeeping, the items that are referred to must be explicitly represented. Finally, the repertoire of possible kinds of observations that can be made remotely is necessarily more limited than for a human observer. Thus, an overall observation may need to be represented by many individual noted facts (positive and negative).

Some investigations will require examining many computers. This will cause an additional level of complexity in the process and in the data collected. The shear scale of the data collected in such investigations may also pose a challenge to investigators.

Thus, it seems that there is a need for an integrated system that can handle a large number of observations and that can organize these in ways that respect the structure of the investigative hypothesis. Thus, to the extent possible, the tool should reconstruct the subtle interplay between "small" observed facts and the "large" hypothesized activities.

### 3.1.3 Impact of Investigation on Critical Infrastructure

Networks and computers commonly perform mission critical functions within organizations, often intermixed with less important activities. Current computer forensic practice will disrupt any *legitimate* activity that might have taken place on those computers, including mission critical ones. Thus, the cost(s) of such an investigation in disruption and inconvenience might easily outweigh its expected value to the organization.

Such disruptions (and their costs) will not be entirely avoidable. Yet, there seems to be a clear need for a lighter-weight form of investigation that can coexist with an operational computer network, and which will have a relatively low impact on users.

## 3.2 Non-disclosure

Security, especially non-disclosure, will certainly be an issue for future network investigations as it is now for nearly all investigations.

### 3.2.1 Keeping Suspects in the Dark

Investigators will usually not want to reveal their activities to suspects, and thus alert them. So, the investigative process should be "silent" on the target computer, at least as far as ordinary users are concerned. The state of the ongoing investigation, and the current investigative activities (including communications) should be well protected, so that they are not very noticeable, and so that they are not comprehensible if they are discovered.

### 3.2.2 Legitimate Privacy Concerns

Investigations may also raise privacy issues since the activities of potentially innocent suspects, of victims, and of bystanders may be revealed in great detail. An investigation (or activities enabled by investigative machinery) could violate privacy in several ways:

- abuse by legitimate investigators,
- outsiders breaking in and making use of the privilege inherent in investigative tools, and
- information inadvertently revealed in a poorly protected investigative record.

A combination of different security strategies will probably be necessary for any integrated network investigation system to mitigate these concerns. Our approach is to use identification and authorization based on strong cryptography to limit access to legitimate users. Those legitimate users are constrained in their activities by various accountability measures. The investigative record, the persistent internal state of

OmniSleuth, and any internal communications are encrypted for privacy, thus limiting opportunities for this information to "leak out."

## 3.3 Integrity

### 3.3.1 Integrity of the Investigative Record

Ultimately, an investigation may cause someone to be accused of wrongdoing. If the evidence and the record of the investigation can be cast into doubt, the whole accusation may be dismissed. Investigators are not likely to have significant doubts about the integrity of findings – it is the accused who will want to raise such doubts. Thus, measures that enhance integrity are seen primarily as defenses against such doubts.

The *completeness* of the investigative record is also an integrity issue. The investigator must be able to defend himself against any allegation that exculpatory evidence was deleted. Our approach in OmniSleuth is to use a combination of techniques: Appropriate bookkeeping makes any *consistent* modification of the record difficult. Digital signatures make the forgery of individual records difficult.

### 3.3.2 Preservation of Raw, Un-interpreted State

Operating systems (and information systems, more generally) present their state to users in the form of abstractions that are much more comprehensible and useful than the underlying representations would be. For example, personal computer users typically deal with *files* without any need to understand the layout of bits on the mass storage device that correspond to that file. Recent computer-forensic practice requires looking "under the hood" for two reasons. First, there is the trustworthiness of the system that is presenting the abstraction: Is it faithful to what is really there? Will it corrupt the underlying representation, destroying the evidence? Since these infrastructural pieces (such as the operating system) are generally complex, and since any software that belongs to a target of the investigation is automatically suspect, it makes sense to preserve and examine the underlying representation.

The second reason that the underlying representation is important is that it may contain information that would otherwise be hidden. For instance, recently deleted files, and even recently used pages of memory, may often be found on the disks of personal computers, though they would not be visible as files.

On the other hand, most of what investigators look at, most of the time, is the abstract form of the data, not the raw bits. However, they may employ trusted software to establish and maintain those abstractions (e.g. a special version of the operating system) thus controlling the threat that the suspect has hacked the operating system (or other infrastructure). How can this paradigm be extrapolated to distributed network investigations? Either parts of the infrastructure must be trusted (and trustworthy) or trusted tools must be deployed to the remote hosts, as needed.

As will be seen, OmniSleuth employs a combination of these strategies, but cannot completely trump the trust issue. The reason for this is that, ultimately, OmniSleuth relies on the host operating system to manipulate fundamental resources such as disk and network adapter. It does not gain *direct* access to the underlying representation, so there are some aspects of host state that will not be accessible via OmniSleuth. If the host operating system were damaged or hacked (or incorrect), then incorrect results could be generated.

OmniSleuth also has an unusual strategy for *preservation*. It does not (and cannot) preserve the underlying state that it has "sensed" – it does not even have access to this. Why did we want to preserve this in the first place? We wanted to preserve this underlying data so that we could convince any skeptics by letting them repeat our analysis from scratch. If there were some other way to convince these skeptics, that should fill the bill as well. OmniSleuth uses trusted software to make its measurements, thoroughly documents all of the steps taken, and preserves this record with high assurance of integrity. It would be very hard to argue plausibly that one of these measurements was not what it appeared to be.

### 3.4 Availability – Ensuring Timely Investigation

Conducting timely investigations could be important because they may prevent further damage or crime. This is one of the advantages of an integrated system for network investigation – since some remote investigating can be done very swiftly. This can even help to focus the more arduous phase of the investigation (e.g., conventional computer forensics) more quickly. However, if a swift network investigation is part of the organization's strategy for handling crime, then that strategy is vulnerable to any attack which impedes such an investigation by making the investigative tools unavailable.

Thus, availability of OmniSleuth is a security issue. There are several plausible denial-of-service attacks against OmniSleuth. The prototype design does not contain countermeasures for these attacks, though such countermeasures should be feasible. These vulnerabilities, and potential countermeasures are discussed in Section 7.3.

## 4 OmniSleuth Capabilities

We construed network investigation as the process of examining (and extending) the state of networked computers to discover traces of the activities of suspects, where this activity is carried out and controlled using the network itself. In this research effort, we restricted our scope to situations in which the networked computers are under the control of the investigating organization. In Section 3, we discussed some of the primary driving forces that, we believe, will shape any successful technology for enabling such investigations, and we alluded to the approach that we have taken with the OmniSleuth prototype in handling these issues. The following sections describe the key capabilities of OmniSleuth that support network investigation and discuss the support for each capability in our prototype:

- distributed search,

- specialized forms of observation,
- organization of findings,
- increasing statefulness, and
- alternative views of data.

## 4.1 Distributed search across many hosts

Many investigations will require widespread searches across many computers. The search may be for particular kinds of traces or for particular vulnerabilities. Such searches have the potential to be expensive, time consuming, and disruptive. They might also result in massive violations of privacy that would be inappropriate, except in the most serious cases. (One is reminded of some recent conventional investigations in which *every* potential criminal in a small community was DNA-fingerprinted in order to solve a serious crime. See [Tag00].)

The key capability of OmniSleuth in support of such widespread search is the ability to directly observe the state of remote hosts. The cost-per-host of such examinations is reduced, making a widespread search feasible in some circumstances where it otherwise might not have been.

Sensing the remote host state is perhaps the most fundamental capability of OmniSleuth, useful even when an investigation is focused on just one or a few machines. The key mechanism in support of this capability is the deployment of agents to remote hosts. The agents, which run in the target environment, have access to a great deal of state information, via ordinary operating system services. A few extra services, introduced when the agent execution environment (called the Agent Guard) is installed, give agents even more "sensory" opportunities.

As agents get more specialized in the kinds of observations they can make, there is the possibility that investigations can be conducted with a reduced impact on privacy. For instance, if an agent is tasked to find and report on any occurrence of a specific file on the remote computer, then the investigator will only learn about that particular file, and the investigative record will substantiate his claim that he did not "snoop around" unnecessarily.

## 4.2 Specialized forms of observation

An investigator might be interested in some very specific bit of information, e.g. what version of Netscape is installed? Such questions are typically answered by combining more primitive observations concerning registry entries, files, and so on. However, there are several advantages to "packaging up" this group of observations plus the associated logic into a new specialized kind of observation:

- It has the potential to reduce unnecessary violations of privacy (as explained in Section 4.1).
- It could reduce network bandwidth required, by reporting just the aggregate observation.

- It could speed things up by making conditional and parameterized observations that depend on one another. For instance, find a registry key containing a file path; then, if the key is found, find the size of the named file.
- It can encapsulate arcane knowledge in an easy-to-distribute, easy-to-use form.

For OmniSleuth, "kinds of observations" equates roughly to "kinds of agents." The key design features of our prototype which support specialized observations are the deployment of observational capabilities in the form of agents as needed, and the support for easily creating new kinds of agents.

We envision that investigators will have a wide variety of agents at their disposal, including some generally useful ones that would be employed in most investigations, and a great many specialized ones that would be used less frequently. It would certainly be impractical to *predeploy* all of that specialized observational capability to a large set of remote hosts, and to maintain it on those hosts. Thus, the deployment, as needed, of observational capability in the form of agents is essential to providing specialized observations.

New agents are written for OmniSleuth by subclassing the *base agent* class, and (typically) overriding a few virtual methods. Thus, authors do not need to comprehend and re-implement the shared functionality of all agents, such as the communications protocol with the Agent Guard. In particular, the "event" abstraction is used by all agents to report their observations. This architecture greatly eases the burden of writing new agents, and would even make it practical to write one specially for an investigation in progress. In the end, this ability to write new agents easily will be essential in producing the large ensemble of different agents that investigators will need. (See Section 5.3)

### 4.3 Organization of Findings

One of the key functions of OmniSleuth is to help the investigator organize evidence. We anticipate that such assistance will be necessary because of the expected size and complexity of evidence. In order to be helpful, the system should support structuring the data in ways that correspond to the investigator's reasoning process. In our view, this (informal) process consists of:

- Decomposing the hypothesis about the crime into a hierarchy of interrelated (or competing) parts.
- Gathering and weighing the evidence for and against each part separately.
- Establishing or refuting a part of the hypothesis should have consequences for the parent(s) of that part, and ultimately, for the whole hypothesis.
- Work on this activity intermittently and non-linearly. Thus, the process must preserve multiple threads of reasoning so that each of them may be put aside and picked up again efficiently.

The principal capabilities of OmniSleuth in support of organizing the findings are the *Theory of the Crime Interface (TOCI)* (Section 5.5), including the persistent storage of its state, and the event database. The TOCI represents the investigative state as a tree of sub-hypotheses. The interface displays the tree in outline form, with the standard expanding/collapsing paradigm familiar to most users of PCs. At the leaves of the tree are atomic assertions about what might have happened. The parents of these nodes have types that include the Boolean connectives *and* and *or*. Thus, the overall hypothesis has roughly the structure of a formula of propositional logic. Each node may have a status of *proved* or *refuted*, which is displayed graphically. Each node may have a collection of *events* associated with it. Events are the fundamental unit of recorded observation within OmniSleuth, and are stored in the event database (Section 5.2).

## 4.4 Increasing Statefulness

*Increasing statefulness* is an important strategy for handling short-lived state and events as described in Section 3.1.1. The primary capabilities of OmniSleuth that support increasing statefulness are:

- persistent agents that can watch for events and can poll the state on a regular schedule,
- the Agent Guard *event cache*, and
- the *query* mechanism.

Observations made by agents (whether immediate, scheduled, or event-driven) are converted into *new* long-lived state when they are converted into *events* and stored in the event cache in the Agent Guard. Since this is a completely local activity, it is relatively inexpensive in terms of computing resources. (This holds for all agents whose cost to make a direct observation is low and which generate events of modest size.) Events in the cache persist until the cache is cleared and can thus outlive the agents that created them. (See Section 7.2 for further discussion of this point.) When the investigator sends a *query* to the Agent Guard, all events from the cache that match the conditions in the query will be returned to the investigator. Thus, the significant costs of network transmission and cryptography for an event are deferred until some interest has been expressed.

The other major issue for increased statefulness is maintaining assurance of integrity between the time of the observation and the time at which the investigator receives the event. This assurance rests, to some degree, on the integrity of the Agent Guard process and the separation of processes enforced by the operating system (which separation is not absolute). New events are immediately transmitted to the Agent Guard and stored in process memory. When the Agent Guard process is later seen to be healthy and functioning normally, one can reasonably infer that its internal data structures (including the event cache) are intact. Other cross-checks that reflect necessary interrelationships among events can further increase confidence that an event is intact because they would make forgery more difficult.

As an event leaves the Agent Guard, it is signed by the cryptographic token on that host. Verification of the signature on the event should give a high degree of assurance that no tampering has occurred from the point in time when the event is transmitted from the agent guard onwards.

## 4.5 Alternative Views of the Data

Visualization facilities within an integrated investigation system will help to handle the expected scale and complexity of the evidence in a network investigation (Section 3.1.2). Such facilities will assist the investigator to comprehend observations and the relationships between observations.

OmniSleuth incorporates a *Timeline Tool* as a visualization aid. This tool places a set of events on a timeline so that their temporal relationship can be easily understood. Each observation made by an OmniSleuth agent is timestamped using the local system clock. However, many kinds of observations will naturally contain other *times* embedded in the data for the event. For instance,

- the observation of file properties made by the File System Agent includes the mod-time and creation-time of the file.
- Any agent which referenced a registry key could also collect the mod-time or creation time of that key.
- An agent may be created to report on things that are themselves timestamped, such as the entries in the system log file.
- Some observations may have a temporal extent. For instance, an agent might directly observe a login session, recording its start- and stop-times.

Temporal relationships between events are important because they often provide a general means to corroborate or disprove *causal* relationships. For instance, the hypothesis may require condition *A* in order to enable the event that causes condition *B*. In this case, condition *A must* be true before condition *B* becomes true. If observations can confirm or rule-out this temporal relationship (or make it less likely) then they will tend to confirm or refute the hypothesis.

OmniSleuth itself is subject to the same forms of causality. Thus, all of the times recorded within an event should be no later than the timestamp of the event itself in order to be consistent. The events that record the start and stop of an agent should bracket the timestamp of all of the events generated by the agent. Those events should have monotonically increasing timestamps. The events recording the start and stop of an Agent Guard process should bracket the events of all of the agents for that AG process. Thus, the consistency of temporal relationships is an important aspect of the designed integrity mechanism of OmniSleuth. The Timeline Tool allows these relationships to be visually inspected.

## 4.6 Transition to Conventional Investigation

We envision that many network investigations will precede or accompany a conventional computer-forensic investigation. Evidence developed via network investigation will narrow the focus of the investigation, thus enabling a more cost effective use of the conventional computer-forensic techniques.

It may be that an ongoing conventional investigation turns up a suspect. In order to continue the investigation without alerting the suspect, network investigation techniques (such as those described here) could be employed. Thus, a smooth interplay between conventional and network investigation is desirable.

The key capabilities of OmniSleuth that contribute to the transition to a conventional investigation (or more generally, to the interplay) are:

- preserving the investigative record with high integrity,
- bookkeeping to preserve the meaning of observations, and
- self-contained reporting on the state of the ongoing or completed investigation.

The first of these points has already been discussed.

The meaning of an observation is tied up in exactly how the observation was made. Consider, for example, a nominally "exhaustive" search for a file. The strongest *meaning* of the result (suppose that nothing was found) is that the sorts of things *that this particular search would have found* were not there. Therefore, we must keep track of the search (or observational method). Our approach to this involves a careful accounting which should allow us to trace each observation back to the agent executable (program) that was used to make it. In support of this, we record observations that contain information about how OmniSleuth was operated. We have also introduced unique IDs that refer unambiguously to Agent Guard processes (instances), agents, and events.

Reports that summarize the state of the investigation can be useful transition documents when a network investigation is "handed off" for a more conventional investigation. Such reports should be complete enough to contain their own supporting data so that they may stand on their own with little reference to external data.

## 5 Architecture

OmniSleuth is an integrated, agent-based system for network investigation. This prototype is based on an earlier agent framework, the *Secure Intrusion Detection Framework (SIDF)*, developed at Odyssey Research Associates under the direction of AFRL Rome Site. OmniSleuth enhances and adapts this framework for use in investigation.

This section describes the architecture of OmniSleuth emphasizing those aspects that were developed on this project. More detailed information about the original SIDF architecture is available in the documentation of that project [Rei97,98A-D].

Due to the limited level of effort on this project, some aspects of this architecture are not fully implemented in the prototype. However, none of these deficits is believed to represent a significant technical challenge; those features are not essential in demonstrating feasibility of the concept of integrated network investigation. Section 7.1 describes the software development results. A detailed discussion of these gaps may be found in the OmniSleuth User Manual [Rei00].

The next subsection gives an overview of the major components of OmniSleuth. Following that are several subsections concentrating on features of the OmniSleuth architecture that were introduced during this effort.

## 5.1 Major OmniSleuth Components

The fundamental capability of OmniSleuth, inherited from SIDF, is the ability to securely launch a new agent (program) on a remote host, and to communicate with that agent and control it. The agent hosting environment on the remote host is the *Agent Guard (AG)*. This is an application or a service (a process), which receives a new agent over the network, starts the agent running, and mediates all communication with agents. In particular, all *inter-host* communication is sent via the AG, which signs and encrypts it. The *Deployment Center (DC)* is the application used to launch and control agents. The DC sends agent executables out to the AGs, and can send and receive a variety of messages to control and monitor agents and AGs. For OmniSleuth, several features were added to the DC user interface that are more specific to investigation. Collectively, these features are referred to as the *Investigative User Interface.* OmniSleuth relies on PKCS-11 compliant cryptographic tokens for digital signatures, for encryption, and to preserve certain critical non-volatile state.

**Figure 1. Agent Deployment in OmniSleuth**

Figure 1 depicts the key steps in deploying an agent. Upon request, an agent from the agent database is signed (with the DC private signing key) and encrypted (with the particular AG's public encryption key). The encrypted file is transmitted to the AG, which decrypts it and verifies the signature. If these operations are successful, the AG starts the agent process running. The AG establishes a persistent local TCP connection to the agent process, and transmits to the agent some identity information.

Once the agent is running, control messages can be sent from the DC to the AG (signed and encrypted, as before), and from there to the agent (in clear text). When the agent has an observation to report, it constructs a timestamped data structure, called an *event*, which is serialized and sent to the local AG, where it is stored in the *event cache*. The event may be sent from the event cache to the DC, either immediately, or upon request. If so, it is transmitted in signed, encrypted form.

## 5.2 Event Subsystem

The *Event Subsystem* is the mechanism that supports the shared *event* abstraction. This, in turn, allows all agents, new or old, to report their observations in a completely uniform way. Events are sufficiently general to encompass just about anything that an agent might have to report. At the lowest level, the event subsystem consists of software for handling some "lisp-like" data structures. The event generation module, the event cache, the query mechanism, and the DC facilities for storage and display of events are all built on top of this layer.

SIDF contains a mechanism to introduce new kinds of messages between entities. However, in OmniSleuth, new messages *associated with a new agent* are not introduced and handled in this way, because this would require that all Agent Guards be recompiled in order to introduce a new message. In OmniSleuth, agents use a fixed set of messages to report their observations and for control purposes. The messages that report observations are very general and flexible, so that they can handle any observation of any agent. The key to this flexibility is the *event* data structure for representing observations.

OmniSleuth employs a very general "lisp-like" data structure to represent observations. These are linked data structures, consisting of pairs (or "cons cells"), integers, strings, and a few other basic data types, together with some distinguished atoms such as *nil*. These data structures can be easily serialized into parenthesized expressions. The serial form (string) can be deserialized back into the linked structure. This serial form, when properly indented in a multi-line style, is also quite readable as a "tree."

An agent that has an observation to report will begin by creating a new event. This is a lisp structure containing a representation of the current time, the agent's unique ID, and the serial number of this event (within the sequence of events for this particular agent). The agent can then add the data representing observed values to this data structure before sending it to the AG. The lisp structure lends itself to flexibility, because within the structure values can be interspersed with identifiers that pin down the meaning of the values.

Events are serialized, sent to the local AG, and deserialized. Events that arrive at the AG *may* be transmitted immediately onwards to the DC if the agent sets a flag in the message. All events, regardless of this flag setting, are saved in the AG event cache. This cache represents the recent history of observations of *local* agents, together with some events that record the lifecycle of agents and of the AG itself.

The investigator may issue a *Query* to an AG, requesting all events that meet some set of criteria. The AG responds to a query by transmitting each matching event (in a separate message) back to the DC. Events that arrive at the DC are placed in the *event database*, where they may be viewed, sorted, and selected for analysis.

### 5.3 OmniAgent, base class

New agents are written for OmniSleuth by subclassing the OmniSleuth Base Agent (or OmniAgent). The OmniAgent is a fully functional agent, with all of the necessary machinery, but which does not make any useful observations. When this base agent is subclassed, some virtual methods may be overridden to provide the observational functionality. The suggested methods to override are:

- *ExecuteCommand* which is used to define the agent's response to command strings that may be sent by the user,
- *FormCreate* which is the appropriate place to initialize most data structures and could be used to make an immediate observation, and
- *HandleTimer* which is used to make observations on a regular schedule.

## 5.4 Notions of Identity for the AG's and the Agents

Some of the integrity of the investigative record is based on the ability to cross-check redundant and related records. Unique names for the AGs, agents, and events are the basis for the internal consistency that makes this possible. Thus, we can associate the event that marks the start of an agent guard process, the event that records the start of a new agent, and an event generated by that agent via their unique identifiers. Uniqueness allows us to cross-check long after the entities involved have terminated.

The agent guard cryptographic token is given an identity consisting of a random sequence of bytes. (With very high probability this sequence will not appear in any other token.) This identity is stored on the token, the first time the token is used.

The token contains a record of a monotonically increasing sequence number. When the AG starts running on a host, this counter is incremented. A combination of the token ID and this sequence number is then used as the unique ID of this AG process. Each time a new agent is launched, the counter is again incremented. The combination of the token ID and this sequence number is used as the unique agent ID.

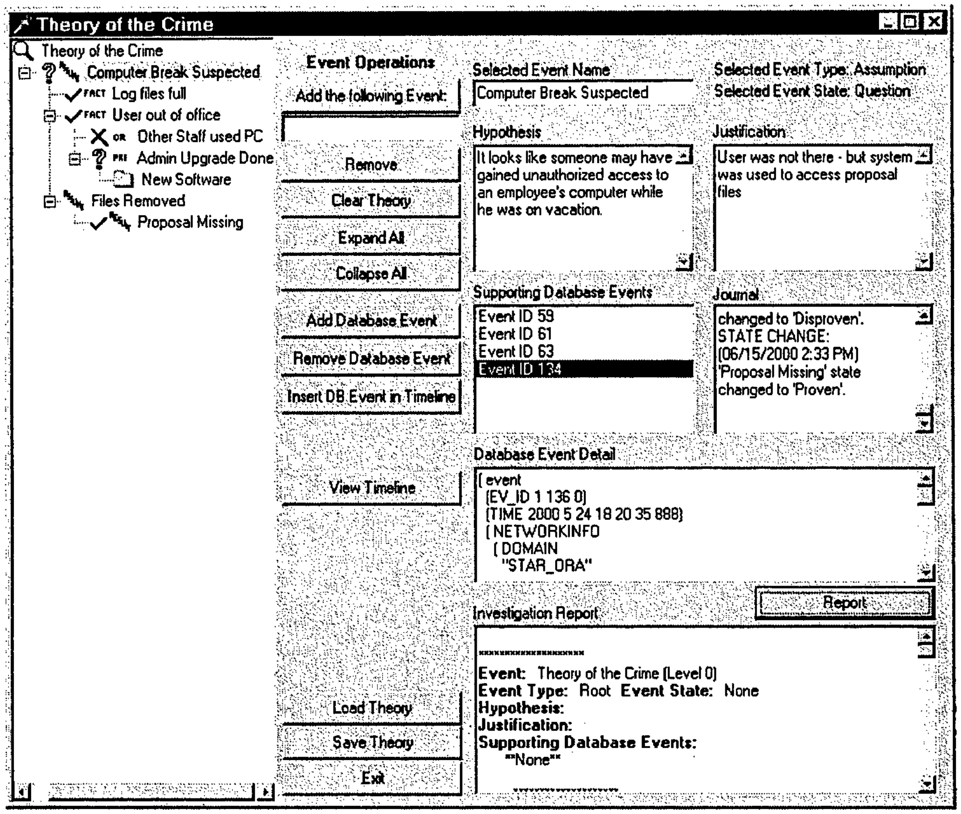
All agents maintain an event counter internally. Each time an agent generates a new event, the counter is incremented, and the count is used as part of the unique ID of the event, together with the agent's ID.

Thus, the unique ID of an event places that event within a sequence of events generated by a particular agent, hosted by a particular AG, running on a particular machine.

When an investigator issues a query to an AG, the response will be a set of events from the cache that match the query (or, possibly, a message saying that there are no such events). OmniSleuth associates a name called a *QueryID* with this entire transaction and uses that QueryID to label all of the messages between the AGs and the DC that are related to one another. Thus, at the DC, the investigator can associate events that are received (or a denial message) with the query that caused those events to be transmitted.

## 5.5 Theory of the Crime Interface

The *Theory of the Crime Interface* (TOCI) is part of the user interface of the Deployment Center. This is the window that investigators will use to organize their findings as the investigation progresses. This window displays (or can display) most of what constitutes the "state" of the investigation. Figure 2 shows the appearance of the TOCI.


Theory of the Crime
Theory of the Crime
Computer Break Suspected
FACT Log files full
FACT User out of office
OR Other Staff used PC
PRE Admin Upgrade Done
New Software
Files Removed
Proposal Missing
Event Operations
Add the following Event:
Remove
Clear Theory
Expand All
Collapse All
Add Database Event
Remove Database Event
Insert DB Event in Timeline
View Timeline
Load Theory
Save Theory
Exit
Report

Selected Event Name: Computer Break Suspected

Selected Event Type: Assumption

Selected Event State: Question

Hypothesis: It looks like someone may have gained unauthorized access to an employee's computer while he was on vacation.

Justification: User was not there - but system was used to access proposal files

Supporting Database Events:
- Event ID 59
- Event ID 61
- Event ID 63
- Event ID 134

Journal:

changed to 'Disproven'.
STATE CHANGE:
(06/15/2000 2:33 PM)
'Proposal Missing' state
changed to 'Proven'.

Database Event Detail

```
( event
  (EV_ID 1 136 0)
  (TIME 2000 5 24 18 20 35 888)
  ( NETWORKINFO
   ( DOMAIN
     "STAR_ORA"
```

Investigation Report

```
******************
Event:  Theory of the Crime (Level 0)
Event Type:  Root  Event State:  None
Hypothesis:
Justification:
Supporting Database Events:
     **None**

    ******************
```

**Figure 2. The Theory of the Crime Interface**

The TOCI has direct access to the other internal data structures of the DC, such as which agents have been deployed and which Agent Guards are "alive." At the moment, the only DC internal data that the TOCI uses is the event database, which contains all of the events received by the DC from various AGs. However, there is the potential for much tighter integration.

The state of the investigation has a tree structure. Each node in the tree represents a part of the hypothesis about the crime. In the current prototype there are seven kinds of nodes defined. The node "kind" modulates its meaning, and also the meaning of its subnodes. Here are the kinds of nodes currently supported, with their intended meanings:

- *OR:* one or more of several alternatives occurred (or was true). The subnodes are the alternatives.
- *AND:* several things all occurred (or were true). The sub-nodes are those things.
- *FACT:* a leaf of the AND/OR tree. This is a part of the hypothesis which is not further subdivided.
- *ASSUMPTION:* a leaf of the AND/OR tree, like a FACT. However, use of this kind of node does not incur a significant proof obligation. It would be used to formalize some aspect of common sense, or common knowledge.
- *PRECONDITION:* describes the necessary conditions under which its parent node *could* have occurred. Multiple pre-conditions of an AND node are considered to be conjunctive. Multiple pre-conditions of an OR node are considered to be disjunctive.

- *POSTCONDITION:* describes the necessary consequences of the occurrence of the parent node event or condition. Multiple post-conditions of an AND node are considered to be conjunctive. Multiple post-conditions of an OR node are considered to be disjunctive.
- *None:* default kind of node, with no special meaning.

The idea behind the pre- and post-condition nodes is to be able to represent structurally the cause-and-effect relationships between parts of the hypothesis. Also, they can be used to keep track of important proof obligations (e.g. to establish that a hypothesized pre-condition was true) and opportunities for corroboration (i.e. to verify that the post-conditions became true).

Each node has a status, which is one of the following:

- *Proven*
- *Disproven*
- *Question*
- *None*

In addition, the nodes have several properties which take string values:

- *Name*
- *Hypothesis:* an explanation of what is supposed to have happened.
- *Justification:* an explanation of the status of this node, including references to the supporting events.

Finally, each node may have an attached list of events called *database events* in this interface. These are the events retrieved from the Agent Guards which the investigator believes will contribute to proving or disproving that node.

The TOCI window shows this tree of nodes in a vertical outline style, using a TreeView control. In this display, the kind and status of the nodes is depicted graphically with icons; the name of the node is used as its primary tag. Parts of the outline may be expanded or contracted (and thus hidden). When a node in the outline is *selected*, then the other attributes of that node are displayed in various panes of the TOCI window. Using this interface, the investigator can easily browse a large investigation and can edit it. All of the properties of the nodes are set manually by the investigator in the current prototype. However, it would certainly be desirable to automate some settings. For instance, the status of a node should be consistent with the status of its children and siblings. Currently, consistency is not enforced.

Other panes of the TOCI show a *Journal* and a *Report*. The journal is a cumulative, timestamped list of edits performed on the theory of the crime. Each time a node is added or deleted, or one of its properties is changed, that is noted in the journal. Thus, investigative actions such as launching a new agent can be placed within the context of an ongoing investigation, providing a powerful form of accountability.

On demand, the TOCI can produce a report, which is a summary of the entire investigative tree, including supporting database events. Such a report could provide the means of preserving or communicating the status of the investigation.

### 5.6 Timeline Tool

The timeline tool is another part of the investigative user interface to the Deployment Center. This is a pop-up window that can display the temporal relationship between several events. Selected events are added to the timeline from either the TOCI window or from the Event Database window. The configuration of events in the timeline is not a persistent feature of the investigation in the current prototype. However, persistence of the timeline, or even attachment of multiple persistent timelines, to the nodes of the TOCI would be desirable.

**Figure 3. The Timeline Tool**

The timeline tool, shown in Figure 3, displays time from left to right in a scrolling window. Each event is displayed as a horizontal stripe within the window, labeled by the name of the event. As explained earlier, an event may have embedded within it multiple times, with the timestamp of the actual observation typically being the latest of these. The horizontal stripe for an event begins at the earliest time in the event and ends at the latest time. Within this strip are marks showing any "interior" times. In the degenerate case when an event has just one time, it is shown as a "mark" without the stripe.

A vertical "cursor" allows the investigator to accurately determine the order of times associated with different events. He can also read the date and time of the cursor position.

The zoom feature allows the scale of the display to be varied from roughly one century to milliseconds. The order of events (vertically) within the display can be adjusted to place related events next to one another.

As mentioned earlier, temporal comparisons can be very important because they can often test (hypothetical) causal relationships. As a very simple example, the mod-time of a file should certainly be in the past. Therefore, an event recording an observation of a mod-time should have a timestamp that is after the mod-time itself.

## 6 Scenario

We developed a scenario of a typical network investigation in order to focus our design efforts, and then to illustrate some key concepts of network investigation, as we envision it. Using the OmniSleuth prototype, we have developed a demonstration based on selected parts of the scenario. This section discusses the scenario and its relationship to key capabilities of OmniSleuth.

The underlying story of the scenario is the investigation of a possible racist gang on an Air Force base. Specific racist documents have been found thus sparking the investigation. The original conjecture is that these documents have been downloaded and printed on the base. Later, we learn that an FTP server on the base is serving up the documents in question – we conjecture that the printed documents derive from this source.

## 6.1 The Scenario

Copies of a known racist tract entitled "White is Right" have been found on an Air Force base. This triggers an investigation into a possible racist organization forming on the base. It is conjectured that the document was downloaded and printed on the base. This document is known to exist in several places in the Internet as `wh_right.doc`, a Microsoft Word document. Investigators face the following questions: Who is involved? What are they up to? *Notes on using this scenario as the basis of a demonstration are in italics, below.*

1. The investigators launch an agent to detect any file with that name, to all machines on the base that are instrumented. These agents will report immediately and persist. They will recheck once per day. Notice that this is a fairly general purpose agent type. *Demonstration would require several agent guarded computers as the targets of this search.*

2. The investigators launch agent to look in web browser cache and favorites (IE) or bookmarks (Netscape) for all users with profiles on those machines. Look for that file, or for any of the IP's known to have it. Note that this is a very special purpose agent – it knows all about the Netscape cache structure for instance. *Current capability is limited to detecting which browsers are installed. May need to enhance the Browser Agent. Can use the DIRLIST command of the FSAgent to find out which users have profiles.*

3. The document is found, but not in a user directory. It is in a directory served by a legitimate FTP server with anonymous access enabled. Notice that this is a surprising result – modify the Theory of the Crime. *We need to establish such an FTP server, with these properties. We need to build an initial Theory of the Crime, in order to modify it.*

4. The investigators launch an agent to look at the containing directory (or redirect the existing agent). They discover that the directory is world writable – an obvious mistake or the result of an intrusion. They save the event in appropriate part of the Theory, together with supporting events. *We can use the FILEINFO command of the FSAgent. What are the supporting events? They are: launching the AG, launching the FSAgent, and sending the command to the agent. If available, termination events for the agent and agent guard would also be relevant.*

5. The investigators look at the FTP log file to see who put the file there and who has downloaded it. But, suppose it is not available for some reason. Perhaps the *FTP squatters* have deleted it. Perhaps logging is not enabled. *Look for the log, using FSAgent, in known places, but don't find it.*

6. The investigators launch an agent to monitor FTP traffic to the site and record source IP and time(s). Also, they task the file system agent to monitor the whole FTP site for any changes to content or access time, say, every 15 minutes. *TCPAgent does not quite record this precise information. However, it could do so. File system agent does not record changes, but it could do so.*

7. When agent shows that the file in question has been accessed, the investigators place that event, together with the FTP traffic log on a timeline, to sort out which IP the request came from. They refine the theory to describe this particular source IP, and save the relevant events there. They print out a report summarizing the investigation and evidence collected so far.

8. If the source IP is an Air Force computer, then seize it and analyze the disk, looking for traces of the file. Notice the transition to a more "conventional" computer forensic analysis.

### 6.2 Testing Our Assumptions Against the Scenario

Let's first look at the assumptions about investigation in future network settings, as applied to this hypothetical situation:

- *Preservation of Raw, Un-interpreted State.* The idea is that we will be able to manage without preserving the raw bits of state – just preserving observations of that state. Certainly, in the early stages of the investigation, this seems quite reasonable, representing an appropriate trade-off between *eventual* assurance of integrity and such other factors as cost and stealth.
- *Ephemeral State and Events.* In the scenario, we envision that being able to monitor FTP traffic (packets) and possibly short-lived files in the FTP directory will further the investigation.
- *Scale and Complexity of the Evidence.* If the search for the files in question is broad, we can envision a very large number of agents launched, queries issued, and responses received. However, this scenario does not require particularly subtle reasoning.
- *Security.* This scenario involves a conjectured on-going crime. Investigators must avoid alerting the suspects. The wide-spread search also has the potential to violate privacy of bystanders – this must be avoided.
- *Impact of Investigation on Critical Infrastructure.* The computers and network in the scenario are part of the infrastructure of the Air Force base. It would be unreasonable to take these computers off-line (e.g. to sequester and examine their hard drives) without a very good reason – a very high likelihood that incriminating evidence will be found. Thus a non-disruptive network investigation seems appropriate as the first stage, in order to identify hosts for more intensive scrutiny.

## 6.3 OmniSleuth Capabilities Used in the Scenario

We earlier identified several capabilities which seem critical for successful network investigation. We will now consider each of these, in turn, to see how they impact the scenario.

- *Distributed search across many hosts.* In the scenario, the investigator searches across many hosts for the specific file, by name. The idea behind this search is that if a file has been casually downloaded there is a good chance that the name of the file was not changed. If the OmniSleuth AG has been widely enough installed, then this kind of search would be feasible, and certainly easier than a manual examination of each machine. However, there are two senses in which the OmniSleuth mechanism is *not* ideal for this purpose: In order to search across a large number of hosts, a scripting facility or "batch" mode to automate repetitive operations would be very convenient. The current user interface would be somewhat cumbersome for this purpose. Second, launching an agent seems like a fairly heavy-weight mechanism for this one-shot search. A more practical design might implement such very common functionality directly in the AG.
- *Specialized forms of observation.* The investigator uses a specialized agent to examine the web browser caches and bookmarks, to see if any of them mention the file being sought. This agent will examine the contents of specific directories, files, and registry entries, in order to make its observation. In doing so, it will incorporate knowledge of how these web browsers function. These same observations could be made with more general-purpose tools, but with several disadvantages: much more cumbersome because of the need for a series of steps that depend on each other, requiring manual intervention (or possibly scripting) by the investigator, much greater network traffic, and the dependence on the investigator's possession of some arcane knowledge that might be out-of-date. It seems that the ability to deploy specialized investigative functions such as this will be critical to the success of any practical network investigation facility.
- *Organize Findings.* The investigator's original and revised hypotheses are compounds of several interrelated events, involving both conjunction and disjunction. Originally, that someone used either FTP or HTTP to download the file and store it and then, they printed the file. In revised form, the hypothesis is more complex. In either case, the alternatives and parts of the hypothesis are supportable (or refutable) by separate observations. For instance, a trace of the file within the browser history would support the HTTP sub-hypothesis, and would suggest who is responsible (due to the user profile), while the simple presence of the file on a host supports the disjunction (FTP *or* HTTP).
- *Increasing Statefulness.* In this scenario, OmniSleuth is used to preserve FTP network activity and also to monitor changes to the FTP directory. The investigator can examine the record of this ephemeral state after the fact. The records are preserved in the same way that more conventional observations are.
- *Alternate views of the data.* The investigator in the scenario makes use of one "alternate" view of the data – provided by the timeline tool.
- *Transition to conventional investigation.* At the interface between a network investigation and a conventional computer investigation, it may be necessary to

cleanly document the findings of the network investigation in a manner that "stands on its own." In this scenario, this report might be used to justify a search warrant, for instance.

# 7 Conclusions and Recommendations

## 7.1 Accomplishments

We developed an understanding of requirements for network investigation. We believe that any practical system for computer investigation in the near future must meet these requirements in one way or another, in order to be successful. (Section 3)

We explored a particular approach to supporting network investigation – an integrated tool based on an agent architecture and framework. One outcome of this exploration is a set of capabilities that could effectively support a network investigation, and which are feasible to implement via an agent architecture. (Section 4)

We demonstrated the feasibility of the agent-based approach by constructing the OmniSleuth prototype (Section 5). OmniSleuth has a suite of functionality and security features that could plausibly enable a network investigation. We developed a scenario of a "typical" network investigation, designed to have all of the significant features of such investigations. The prototype is capable of carrying out most steps of the scenario. Our *design* for OmniSleuth would be capable of executing the entire scenario. The OmniSleuth software developed includes several example agents that illustrate the kinds of observations that agents can make, and how such agents would be developed.

Specific software development accomplishments include enhancements to the SIDF infrastructure, development of the Investigative User Interface, and development of several example agents. Here is a more detailed list:

- *New message types* for commanding agents (to control observations), for transmission of observations to the AG and DC, and to initialize the state of an agent.
- *Enhancements to the Agent Guard* to handle the arrival of new observations from agents. The newly arrived observations ("events") are saved in the *event cache*, an internal data structure of the AG. The AG also handles *query* messages from the investigator, returning those events in the event cache that match the query.
- *The event data structure,* a very general, extensible data structure to represent the observations of nearly any agent.
- *The Investigative User Interface,* enhancements to the SIDF Deployment Center that are specific to investigation. These include the *event database*, the *Theory of the Crime Interface*, and the *Timeline Tool*, as well as interfaces to issue commands to agents and to issue queries to the AGs.
- *Stronger bookkeeping for accountability and integrity.* These features include strong notions of identity for tokens, AGs, agents, and events, all based on data stored safely in cryptographic tokens. Also, key events in the lifecycles of OmniSleuth components are recorded with sufficient redundancy that effective cross-checking is possible.

- *The OmniAgent,* a base class for investigative agents. This is an enhancement of the SIDF agent base class.
- *Example agents,* including a *TCP Monitor Agent*, a *File System Agent*, a *Browser Agent* and a *Network Information Agent.*

## 7.2 Current Status

The OmniSleuth prototype, including several investigative agents, runs and is sufficiently complete to support a demonstration. This demonstration is based on a scenario of a network investigation (Section 6) and is intended to illustrate the concepts and requirements of network investigation and of our agent-based approach.

Due to the limited size of this research effort, some aspects of the OmniSleuth design are not fully implemented in the prototype. However, none of these represents a significant technical challenge that could affect the feasibility of our approach. Thus, we consider feasibility to be well established. The OmniSleuth Users Manual contains a detailed discussion of these unimplemented aspects.

## 7.3 Future Directions

During the course of our work on OmniSleuth, the many ideas for enhancements (or variations) were considered. The following is a list of those ideas or directions that seem most promising:

- *Scripting* or batch processing for repetitive operations. For instance, launching the same agent to many hosts, or sending a similar command to many deployed agents.
- *Protection against Denial of Service Attacks.* Possible countermeasures include: agile use of TCP ports by AGs and the DC, the ability to *restart* an AG remotely, and, the ability of an agent to "pick up" from where an old one left off (via checkpointed state).
- *Separate Verification Module.* Currently, signatures on messages and databases are verified within OmniSleuth. However, it might be necessary, as part of some legal proceeding to enable other parties to be able to verify signatures, without giving them access to all of OmniSleuth.
- *Key Management.* Current provisions for key management are somewhat crude. Yet, effective key management schemes exist in both the research literature and in industry.
- *Synergy with Intrusion Detection Systems.* This promising idea has not been well explored yet.
- *Finer-grained Agents.* Agents on a host could cooperate with one another to make observations, thus avoiding the duplication of overlapping functionality. Some very common observational capability could be built into the AG.
- *Digital Warrants.* We envision a certificate-based warrant system that would interact with the access control mechanism of OmniSleuth. OmniSleuth would enforce the search limitations specified in the warrant.
- *Enhancements to the TOCI,* especially some more advanced editing facilities.

- *Enhancements to the Timeline Tool,* especially multiple persistent timelines that can be attached to the nodes of the TOCI tree and can be annotated.

## 8 References

[Tag00] Stewart Taggart, DNA Testing Furor in Wee Waa. Wired News, April 18, 2000. http://www.wired.com/news/medtech/0,1286,35727,00.html

[Rei97] Mark E. Reilly, Scalable Intrusion Detection and Response Framework System Specification, Odyssey Research Associates, Inc. Technical Report No. SIDF-SS-TM-97-0032-Rev00-(9712), December 1997.

[Rei98A] Mark E. Reilly, Scalable Intrusion Detection and Response Framework Architecture Specification, Odyssey Research Associates, Inc. Technical Report No. SIDF-AS-TM-98-0002-Rev00-(9803), March 1998.

[Rei98B] Mark E. Reilly, Scalable Intrusion Detection and Response Agent Interface Control Document, Odyssey Research Associates, Inc. Technical Report No. SIDF-ICD-TM-98-0003-Rev00-(9803), March 1998.

[Rei98C] Mark E. Reilly, Scalable Intrusion Detection and Response Framework Detailed Design Document, Odyssey Research Associates, Inc. Technical Report No. SIDF-FDD-TM-98-0004-Rev00-(9804), April 1998.

[Rei98D] Mark E. Reilly and Maureen Stillman, An Open Infrastructure for Scalable Intrusion Detection, Odyssey Research Associates, Inc. Technical Report No. TM-98-0023, 1998.

[Rei00] Mark E. Reilly, OmniSleuth Computer Forensic System Software User Manual, Odyssey Research Associates, Inc. Technical Report No. OMNI-TM-00-0008.

# *MISSION*
# *OF*
# *AFRL/INFORMATION DIRECTORATE (IF)*

*The advancement and application of Information Systems Science and Technology to meet Air Force unique requirements for Information Dominance and its transition to aerospace systems to meet Air Force needs.*